Few-Shot Learning in Long-Tailed Settings

Davis Wertheimer, Luming Tang, Dhruv Baijal*, Pranjal Mittal*, Anika Talwar*, and Bharath Hariharan

Abstract—Traditional recognition methods typically require large, artificially-balanced training classes, while few-shot learning methods are tested on artificially small ones. In contrast to both extremes, real world recognition problems exhibit heavy-tailed class distributions, with cluttered scenes and a mix of coarse and fine-grained class distinctions. We show that prior methods designed for few-shot learning do not work out of the box in these challenging conditions, based on a new "meta-iNat" benchmark. We introduce three parameter-free improvements: (a) cleaner and more efficient episodic training procedures based on cross-validation, (b) novel architectures that localize objects using limited bounding box annotations before classification, and (c) simple parameter-free expansions of the feature space based on bilinear pooling. Together, these improvements double the accuracy of state-of-the-art models on meta-iNat while generalizing to prior benchmarks, complex neural architectures, and settings with substantial domain shift.

1 Introduction

Image recognition models have purportedly reached human performance on benchmarks such as ImageNet, but depend critically on *large, balanced, labeled training sets* with hundreds of examples per class. This requirement is impractical in many realistic scenarios, where concepts may be rare or have very few labeled training examples. Furthermore, acquiring more labeled examples might require expert annotators and thus be too expensive. This problem is exacerbated in applications (e.g., robotics) that require learning new concepts on the fly in deployment, and cannot wait for a costly offline data collection process.

These considerations have prompted research on the problem of "few-shot" learning: recognizing concepts from small labeled sets [1], [2], [3], [4]. This past work builds "learners" that can learn to distinguish between a small number of unseen classes (often fewer than 20) based on an extremely small number of training examples (e.g. 5 per class). However, multiple challenges plague these methods when they are applied to real-world recognition problems.

First, few-shot methods typically assume balanced datasets, and optimize the learner for an exact, often unrealistically small number of training examples per class. In contrast, real-world problems may have highly imbalanced, heavy-tailed class distributions, with orders of magnitude more data in some classes than in others. A practical learner must therefore *work equally well for all classes irrespective of the number of training examples*. It is unclear how or even if few-shot methods can handle such an imbalance.

Second, few-shot learning methods often assume the number of relevant concepts to be small, and as such highly distinct from each other. In contrast, real world applications often involve thousands of classes with subtle distinctions. These distinctions can be particularly hard to detect when natural images are cluttered or difficult to parse (Figure 1, bottom). Thus, the learner must also be able to *make fine-grained class distinctions on cluttered natural images*.

We first evaluate prototypical networks [3], a simple yet state-of-the-art few-shot learning method, on a realistic benchmark based on the heavy-tailed class distribution and subtle class distinctions of the iNaturalist dataset [5].

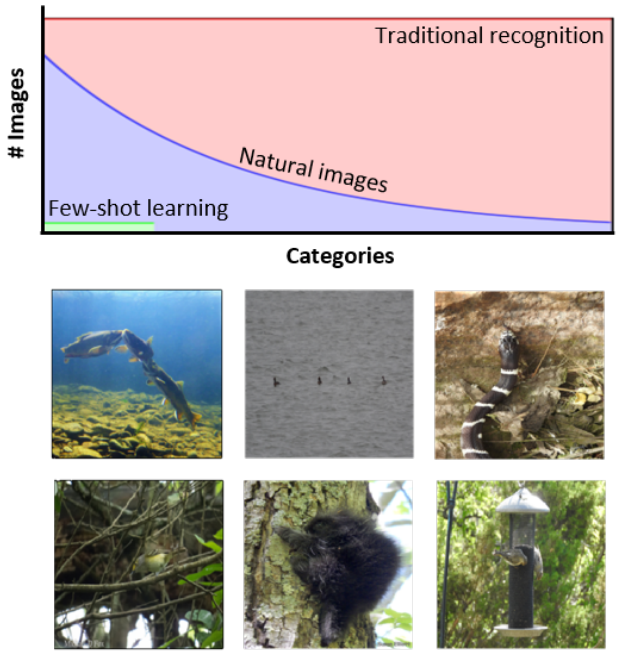

Fig. 1. Discrepancies between existing benchmarks and real world problems. **Top:** Traditional recognition benchmarks use many, equally large classes, while few-shot benchmarks use few, equally small classes. Natural problems tend to be heavy-tailed. **Bottom:** Clockwise from top left: relevant objects may be overlapping, tiny, occluded, underemphasized (bird is on the feeder), blurry, or simply hard to delineate [5].

We show that prototypical networks can struggle on this challenging benchmark, confirming the intuitions above.

We next present ways to address the challenge of heavy-tailed, fine-grained, cluttered recognition. We introduce modifications to prototypical networks that *significantly improve accuracy without increasing model complexity*.

First, to deal with heavy class imbalance, we propose a *new training method based on leave-one-out cross-validation*. This approach makes optimization easier and the learner more resilient to wider distributions of class sizes. It is efficiently implementable as a margin-based loss and yields a **4 point gain** in accuracy.

*equal contribution, ordered alphabetically by last name

Second, we posit that when objects are small or scenes are cluttered, the learner may find it difficult to identify relevant objects from image-level labels alone. To tackle this problem, we explore *new learner architectures that localize each object of interest before classifying it*. These learners use bounding box annotations for a tiny subset of the labelled images. Localization improves accuracy by **6 points**, more so when objects occupy under 40% of the image.

Even after localizing the object, the learner may need to look for subtle distinctions between concepts. Existing few-shot methods rely on the learning process alone to build informative feature representations. We show that straightforward, parameter-free adjustments can significantly improve performance. In particular, we find that *the representational power of the learner can be significantly increased by leveraging bilinear pooling* [6], [7], [8]. While in its original formulation, bilinear pooling significantly increases the model parameter count, we show that it can be applied to prototypical networks with *zero* increase. This modification significantly improves accuracy by up to **9 points**.

Together, these contributions **double** the accuracy of prototypical networks on our challenging heavy-tailed benchmark, with negligible impact on model complexity. Similar gains are observed for deeper networks, varying training schemes, and evaluation settings with fine granularity or large domain shift. These results suggest that our proposed approach provides significant benefits for realistic recognition problems in the wild.

2 Related Work

The ideas behind our proposed techniques have broad prior support, but appear in mostly disjoint or incompatible problem settings. We adapt these concepts into a unified framework for few-shot recognition in real-world scenarios.

Meta-learning: Prior work on few-shot learning has mainly focused on optimizing a *learner*: a function that takes a small labeled training set and an unlabeled test set as inputs, and outputs predictions on the test set. This learner can be expressed as a parametric function and *trained* on a dataset of "training" concepts so that it generalizes to new ones. Because these methods train a learner, this class of approaches is often called "meta-learning". Optimization may focus on the learner's parameterization [9], [10], [11], its update schedule [12], [13], the generalizability of a built-in feature extractor [4], [14], [15], [16], [17], or a learned distance metric in feature space [3], [18], [19]. An orthogonal approach is to generate additional, synthetic data [2], [20]. Recent papers in meta-learning literature have employed several approaches including metric scaling, task conditioning [4], [21], [22], probabilistic meta-learning [14], [23], [24], and even test-time gradient-based adaptation on limited sets of parameters [11], [25], [26]. We focus on prototypical networks [3] in our work, as despite their simplicity these models remain competitive with state-of-the-art when trained or pretrained appropriately [22], [27].

In most cases, few-shot classifiers [3], [9], [11], [12], [13], [15], [16], [18], [21], [25], [26], [28] are evaluated on only a small collection of datasets: mini-ImageNet [4], CUB [29], Omniglot [30], few-shot CIFAR [21], and tiered-ImageNet [31]. These benchmarks present only five classes at a time, with one or five training images per class. Some work has expanded the number of classes [2], [22], but still assumes novel classes have the same number of examples. These benchmarks are therefore divorced from real-world conditions, which can involve many concepts and varying amounts of training data [5], [32], [33]. Many prior meta-learning approaches are incompatible with these settings.

Heavy-tailed datasets: Heavy-tailed class distributions are common in the real world. MS-COCO [34], the SUN database [33], DeepFashion [32], MINC [35], and Places [36] are all examples where an order of magnitude separates the number of images in the most versus the least common classes. MINC and Places are especially noteworthy because they are explicitly designed to *narrow* this gap in data availability [35], [36], yet display heavy class imbalance anyway. Despite this trend, standard recognition benchmarks like ImageNet [37], CIFAR-10, and CIFAR-100 [38] heavily curate their data to ensure that classes remain nicely balanced and easily separable. Most few-shot benchmarks encode class balance explicitly [2], [4], [20], [31].

Improving feature space: It is well known that higher-order expansions of feature space can raise the expressive power of hand-designed feature extractors [39], [40]. Recent work has shown that similar techniques [6], [7], [8] also improve the performance of convolutional networks. The improvement is especially large in fine-grained classification settings, such as facial recognition [8], [41], [42]. However, using the resulting expanded feature space requires parameter-heavy models, even in the few-shot setting [19]. We adapt bilinear pooling [8] as a truly parameter-free expansion, which no longer risks overfitting to small datasets.

Localization: A close relationship exists between localization and recognition. Networks trained solely on image-level, classification-based losses nevertheless learn to localize objects of interest [43], [44]. These learned localizations can act as useful data annotation, including for the original recognition task [44], [45], [46]. Very difficult problems, however, may require expensive ground truth annotations to begin bootstrapping. Fortunately, a very small set of annotations can be sufficient to predict the rest [10]. Semi-supervised localization further improves when image-level category labels are provided [47], [48]. Since each can bootstrap from the other, combining recognition and localization may prove a particularly effective remedy for data scarcity.

Several recent papers have explored more implicit localization techniques, learning to associate object regions without explicit human annotation providing ground-truth correspondence. Techniques for this include generating cross attention maps to highlight target object regions [28], using Earth Mover's Distance (EMD) and a cross-reference mechanism to minimize background noise [25], and utilizing a Transformer architecture that uses spatial correspondence and is robust to domain shift [49]. These implicit localization techniques can be combined with explicit ones and so we consider this line of work orthogonal to ours.

3 Problem Setup and Benchmark

Our goal is to build *learners*, systems that can automatically learn new concepts under challenging real-world conditions, with heavy-tailed distributions of classes and subtle

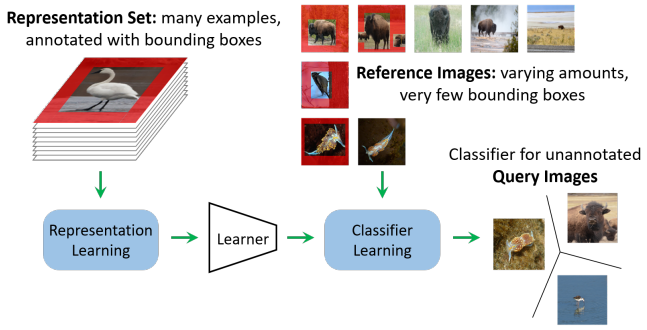

Fig. 2. Our long-tailed, real-world learning benchmark. Initially, many images are available with bounding box annotations. The learner must adapt to new classes using varying but limited amounts of data, with very few bounding boxes. At test time in the wild, there are no annotations.

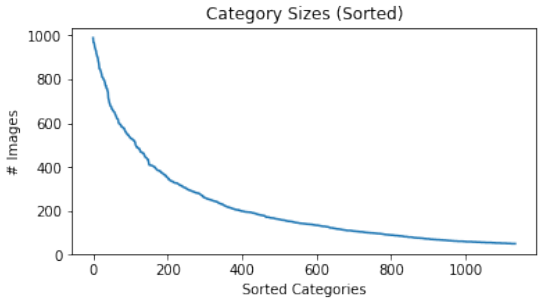

Fig. 3. Class sizes in meta-iNat, demonstrating a long-tailed distribution.

class distinctions. Each learner may have tunable parameters or hyperparameters. As in prior work, these parameters are learned on a "representation set" of concepts ("base classes" in [2]) with many training examples (see Fig. 2).

Once trained, the learner must generalize to a disjoint "evaluation set" of novel categories. The evaluation set is split into a small collection of labeled "reference images" and a larger set of unlabeled "query images". The learner may use the reference images to define the new set of categories, estimate new parameters for those categories (e.g. a linear classifier) and/or fine-tune its feature representations.

Final accuracy is reported on the unannotated query images. We report top-1 and top-5 accuracy, both as a mean over images and over the categories of the evaluation set. The latter metric penalizes classifiers that focus on large categories while ignoring smaller ones.

Two approaches to the above problem act as illustrative examples. A traditional transfer learning approach is to train a softmax classifier on the representation set. On the evaluation set, the fully-connected layer is replaced by a new version with the appropriate number of categories, and fine-tuned on reference images. Query images form the test set. Meta-learning approaches, such as prototypical networks, train a parametric learner on tiny datasets sampled from the representation set, teaching the learner to adapt to novel tiny datasets. The learner processes the evaluation set in a single pass, with reference images forming the training set and query images forming the test set.

Object location annotations: As discussed in Section 1, a key challenge in real-world recognition problems is finding relevant objects in cluttered scenes. Small sets of image-level class labels may be insufficient. We therefore provide bounding boxes for *a small fraction* ($\leq 10\%$) *of the reference images in the evaluation set*. Note that with extremal point clicks, these annotations are cheap to acquire in practice [50]. We fully annotate the representation set, as such datasets tend to be heavily curated in the real world (Fig. 2).

3.1 Benchmark Implementation

We now convert this problem setup into a benchmark that accurately evaluates learners on real-world heavy-tailed problems. For this, the evaluation set must satisfy three key properties. First, as in many real-world problems, training sets should be heavily imbalanced, with orders of magnitude difference between rare and common classes. Yet the number of examples per class must be neither unnecessarily small (e.g. fewer than 10), nor unrealistically large (e.g. more than 200). Second, in contrast to past few-shot learning benchmarks that use five classes at a time [4], [26], [30], [31], there should be many (e.g. at least 20) categories in the evaluation set, with coarse- and fine-grained distinctions, as in the real world. Third, images must be realistically challenging, with clutter and small regions of interest.

We implement our benchmark using the iNat2017 dataset [5], an organically collected, crowdsourced compendium of living organisms, with fine- and coarse-grained species distinctions, a heavy-tailed class size distribution, and bounding box annotations for a significant subset. Of the appropriately-sized categories with bounding boxes, 80% are randomly assigned to the representation set, and the rest to the evaluation set. Within the evaluation set, 20% of images are reference images and the rest are query images, for an overall split of 80/4/16% representation, reference, and query. We propose this "meta-iNat" dataset as a realistic, heavy-tailed, fine-grained benchmark for meta-learning algorithms. Meta-iNat contains 1,135 animal species, the distribution for which can be found in Fig. 3.

While all images in meta-iNat have bounding box annotations, only 10% are allowed during evaluation (see Section 3). We run ten trials on the evaluation set with a different collection of annotated reference images in each trial.

4 Approach

We build upon prototypical networks [3] (Section 4.1) by introducing three light-weight and parameter-free improvements. Batch folding (Section 4.2) improves gradients during training and helps the learner generalize to large classes. Few-shot localization (Section 4.3) teaches the learner to localize an object before classifying it. Covariance pooling (Section 4.4) greatly increases the expressive power of prototype vectors without affecting the underlying network architecture. In addition to being parameter-free, these techniques are mutually compatible and mutually beneficial.

4.1 Prototypical Networks

We briefly review prototypical networks [3]. Prototypical networks are a learner architecture designed to learn novel classes using few training examples. The learner uses a feature extractor to embed labeled reference and unlabeled

query images in feature space. Reference image embeddings are averaged within each class to generate a "prototype" vector for that class. Predictions are made on query embeddings based on L2 proximity to each class prototype.

Training a prototypical network amounts to setting the parameters of the feature extractor, since classification is non-parametric. The prototypical network is trained on the representation set by sampling *small* datasets of reference and query images. These are passed through the network to get class probabilities for the query images. Cross entropy loss on query images is then minimized. Through this training, the network learns a feature extractor that produces good prototypes from limited reference images.

4.2 Batch Folding

Batch folding is motivated by the fact that during training, every image in a batch is either a reference or a query image, but never both. While reference images learn to *form* good class centroids, query images *gravitate toward* the correct centroid and *away* from others. Gradients for both are necessary for learning, but every image gets only one, so prototypical weight updates are noisy.

This reference/query distinction also limits the number of reference images a network can handle. For a prototypical network to work on common classes as well as rare ones, it must be trained with a larger number of reference images [3]. Increasing the reference images per batch, however, requires either increasing the batch size, which runs into memory constraints, or decreasing the number of queries, producing noisier query gradients. Thus the original prototypical networks are *designed* for rare classes.

As an alternative, we propose to use leave-one-out cross-validation within each batch, abandoning the hard reference/query split. The entire batch is treated as reference images, and the contribution of each image is subtracted ("folded") out from its corresponding prototype whenever it acts as a query. Each image thus gets a combined, cleaner gradient, acting as both a reference and a query. Furthermore, the number of query / reference images can be as high as the batch size / one less. The result is stable training with large reference sets without violating memory constraints. We call this approach batch folding.

Procedure: Let n be the number of classes and p the number of images per class in a batch. Denote by $v_{i,j}$ the feature vector of the i-th image in the j-th category. Let $c_j = \frac{\sum_i v_{i,j}}{p}$ be the centroid of the j-th class. To make predictions for the i-th image in the j-th category, the network uses the following class prototypes:

$$c_1,\ c_2,\ \ldots\ c_{j-1},\ \frac{p}{p-1}(c_j - \frac{v_{i,j}}{p}),\ c_{j+1},\ \ldots\ c_n \tag{1}$$

Calculated naively, as in Eq. 1, batch folding is efficiently parallelizable using tensor broadcasting. The necessary broadcast operations are built-in to most machine learning libraries, including NumPy [51], PyTorch [52], and TensorFlow [53]. However, evaluating Eq. 1 still requires computing a unique set of centroids for every image in the batch, which can introduce undesirable overhead. Fortunately, batch folding can be implemented efficiently using an algebraically equivalent rescaling layer. Since we use the negative squared Euclidean distance as our class prediction logit, we can mimic the impact of batch folding by reweighting the logit for the ground-truth positive class j before passing it to the softmax layer. Letting $\delta_{i,j} = -||c_j - v_{i,j}||^2$ represent the unfolded class j logit, the reweighting factor for batch folded logit $\delta'_{i,j}$ can be derived as follows:

$$\begin{aligned}
\delta'_{i,j} &= -||\frac{p}{p-1}(c_j - \frac{v_{i,j}}{p}) - v_{i,j}||^2 && (2)\\
&= -||\frac{pc_j}{p-1} - \frac{v_{i,j}}{p-1} - \frac{p-1}{p-1}v_{i,j}||^2 && (3)\\
&= -||\frac{p}{p-1}(c_j - v_{i,j})||^2 && (4)\\
&= \frac{p^2}{(p-1)^2}\delta_{i,j} && (5)
\end{aligned}$$

Eq. 5 obviates the need to calculate the folded centroids. Instead, simply compute class centroids c_j and logits $\delta_{i,j}$ for $1 \le i \le p, 1 \le j \le n$ as normal. Then, for image embedding $v_{i,j}$, emit reweighted logits $\{\delta'_{i,k} : 1 \le k \le n\}$ where:

$$\delta'_{i,k} = \begin{cases} \delta_{i,k} & k \neq j \\ \frac{p^2}{(p-1)^2}\delta_{i,k} & k = j \end{cases} \tag{6}$$

Since $\delta_{i,k}$ is always negative and $\frac{p^2}{(p-1)^2}$ is greater than one, Eq. 6 effectively pushes the predicted class assignment away from the ground truth. Batch folding can thus be thought of and implemented as a margin, where the classifier must not only predict the correct class, but also to within a threshold of confidence. This margin-based implementation is more simple and efficient than calculating the unique centroids for each embedding as in Eq. 1.

At test time, we eliminate the margin and use $\delta_{i,k}$ directly. In this setting the reference and query images are distinct, so there is no need to batch fold.

4.3 Localization

Image-level labels are less informative when the object of interest is small and the scene cluttered, since it is unclear what part of the image the label refers to. Given many, sufficiently different training images, the machine eventually figures out the region of interest [44]. But with only a few images and image-level labels, distinguishing relevant features from distractors becomes highly difficult.

For these reasons, isolating the region of interest (on *both* reference and query images) should make classification significantly easier. We consider two possible approaches. In **unsupervised** localization, the learner internally develops a category-agnostic "foregroundness" model on the representation set. **Few-shot** localization uses reference image bounding boxes on the *evaluation set* for such localization.

Procedure: In both approaches, the localizer is a submodule that classifies each location in the final 10×10 feature map as "foreground" or "background". This prediction is calculated as a softmax over each pixel embedding's negative L2 proximity to a foreground vector and to a background vector. In unsupervised localization, these vectors are learned parameters optimized on the representation set. In few-shot localization, the localizer gets a few reference images annotated with bounding boxes. We use these boxes as figure/ground masks, and average all the foreground

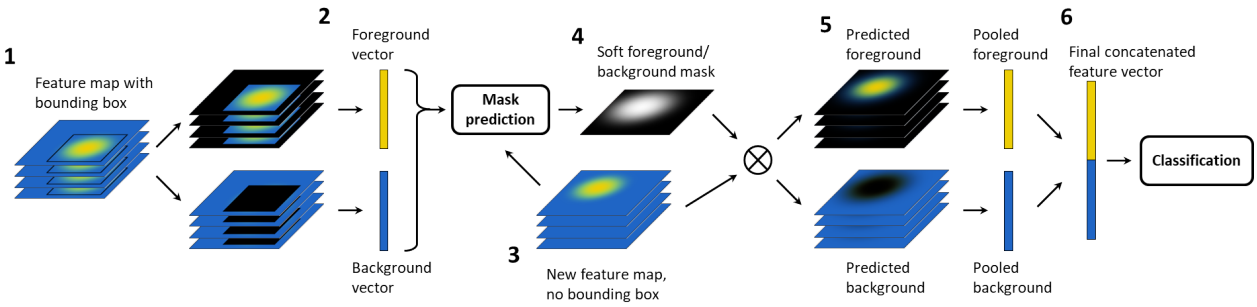

Fig. 4. Few-shot localization. Provided bounding boxes mask off foreground and background regions (1), which are averaged to produce foreground and background feature vectors (2). Pixel features on new feature maps (3) are classified as foreground or background based on distance from those vectors (4). The predicted mask separates foreground and background regions (5), which are average pooled independently and concatenated (6). Unsupervised localization learns the foreground/background vectors as parameters, and begins at (3).

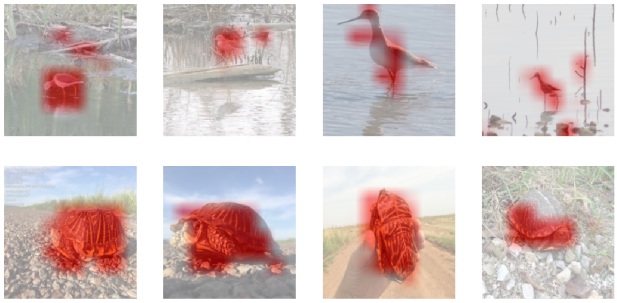

Fig. 5. Example outputs of the few-shot localizer. The leftmost image provides the foreground and background centroids for each row. The network learns without supervision or dedicated parameters to isolate (mostly) appropriate regions of interest.

pixel embeddings to produce the foreground vector. The background vector is computed similarly.

The output of the localizer is a soft foreground / background mask. Multiplying the feature map with its mask (and inverse mask) produces foreground and background maps, which are average-pooled then concatenated. This double-length feature vector is used to form prototypes and perform classification. Fig. 4 provides a visual explanation.

Training: Both localization approaches are trainable end-to-end, so we train them within the classification problem. We use no additional supervisory loss; localizers are trained only to be useful for classification. Despite this, the outputs are visually quite good. Examples are given in Fig. 5.

When a few-shot localizer is trained with batch folding, an additional round of folding during localization is required. Each image's contribution is removed from the foreground and background vectors (as well as the class centroid) when it acts as a query. Otherwise, each query image 'sees' its own ground truth bounding box, preventing generalization to unannotated images.

4.4 Covariance Pooling

For hard classification problems, methods such as bilinear pooling [8], fisher vectors [40] and others [7], [41] can be used to expand the feature space and increase expressive power. Unfortunately, these expanded representations are traditionally sent through linear classifiers or multilayer networks [8], [19], [42], dramatically increasing parameters and making the model prone to catastrophic overfitting.

However, these techniques can be adapted to prototypical networks without any parameter increase. We use bilinear pooling [8],[1] which improves fine-grained classification performance and generalizes many hand-designed feature descriptors such as VLAD [39], Fisher vectors [40], and Bag-of-Visual-Words [55]. This approach takes two feature maps and computes the cross-covariance between them, by performing a pixel-wise outer product before average-pooling. In our localization models, the predicted foreground and background maps serve this role. Otherwise, we use the outer product of the feature map with itself. Both versions perform signed square-root normalization, as in bilinear pooling, but do not project to the unit sphere.

It is worth emphasizing that this expansion adds no parameters. Unlike prior models, all improvement in performance comes from increased feature expressiveness, not from increased network capacity. To emphasize this distinction, we call this version covariance pooling.

5 Experiments

We first present overall results on the meta-iNat benchmark (Table 1), using a variety of algorithmic approaches. Beginning with simple supervised baselines, we show that dealing with class imbalance in meta-iNat is challenging, and that the representation set is indispensable to good performance. We then evaluate representation-learning based approaches, and find that prototypical networks handle class imbalance, but underperform more recent strong baselines. Batch folding, localization, and covariance pooling, however, swiftly overcome this performance gap and improve greatly upon baseline results. We employ a four-layer convolutional architecture closely mimicking [3]. As images are not cropped or centered, we average-pool the final feature map instead of flattening it, removing unwanted spatial priors. Further details can be found in Appendix A.

5.1 Supervised Baselines

To verify the usefulness of the representation set in meta-iNat, we apply straightforward supervised learning approaches directly to the reference images of the evaluation

1. Similar techniques have been called second-order pooling [6], higher-order pooling [7], and covariance descriptors [54].

Model	Top-1 Accuracy		Top-5 Accuracy	
	Mean	Per-Class	Mean	Per-Class
Softmax	13.35±.24	6.55±.19	34.46±.30	20.05±.30
Reweighted Softmax	6.92±.19	7.88±.16	21.94±.31	22.53±.29
Resampled Softmax	1.54±.06	.99±.02	3.77±.01	2.75±.03
Focal Loss [56]	11.96±.13	7.27±.09	32.1±.36	19.34±.21
Class Balanced Focal [57]	11.81±.12	6.91±.08	31.87±.2	20.15±.19
SimpleShot (UN) [58]	18.27±.19	19.98±.21	44.08±.27	43.91±.24
SimpleShot (CL2N) [58]	18.57±.13	20.36±.18	44.54±.27	44.68±.17
Baseline [26]	24.62±.17	13.96±.16	51.72±.07	35.22±.19
Baseline++ [26]	19.52±.59	7.87±.59	46.18±.42	24.75±.43
Balanced Baseline [26]	22.41±.23	20.62±.27	47.59±.28	45.21±.38
Balanced Baseline++ [26]	22.55±.72	16.91±.26	50.89±.31	43.67±.36
RFS-simple [17]	26.21±.21	20.79±.25	53.94±.09	44.44±.24
PN	16.07±.19	17.55±.19	42.1±.21	41.98±.18
PN+BF	20.04±.04	20.81±.08	47.86±.31	46.57±.23
PN+BF+fsL*	26.25±.05	26.29±.04	55.43±.09	53.01±.08
PN+BF+usL	28.75±.13	28.39±.15	57.90±.24	55.27±.37
PN+BF+usL+CP	32.74±.13	30.52±.13	61.32±.14	56.62±.16
PN+BF+fsL+CP*	**35.52±.05**	**31.69±.06**	**63.76±.09**	**57.33±.10**

TABLE 1
Results on the meta-iNat benchmark, with 95% confidence intervals from 4 trials. PN is a prototypical network, BF is batch folding, fsL and usL are few-shot and unsupervised localization, and CP is covariance pooling. *Results are averaged over 10 runs of 4 trials, annotations randomly sampled per-run.

set. As shown in the top rows of Table 1, standard softmax classifiers trained from scratch on reference images perform poorly, especially on rare classes, as evidenced by the low per-class accuracy. We address this bias by reweighting the classification loss during training for each example inversely proportional to the frequency of the class containing that example. Reweighting by class in this way improves the per-class accuracy only slightly, while damaging average performance per-image. Oversampling the rare classes using this same weighting scheme causes catastrophic overfitting.

We consider a classifier trained using focal loss [56] as an alternative baseline. Focal loss reweights the standard negative log-likelihood so as to put more weight on instances that are difficult to classify, while decreasing the contribution of easy, already correct predictions. Since instances from rare classes tend to underperform, the focal loss acts as an implicit class-rebalancing method (for further discussion see [56]). Formally, the focal loss FL on a predicted probability distribution p over classes, with ground truth class c, is:

$$\mathrm{FL}(p) = -(1-p_c)^{\gamma}\ln(p_c) \tag{7}$$

where p_c is the predicted probability for class c and γ is a hyperparameter.

[57] introduces an additional, explicit class-balancing element to the focal loss which we also consider as a baseline. In order to obtain class balanced focal loss, we incorporate a weighting factor $(1-\beta)/(1-\beta^{n_c})$ to Eq. 7, where β is a hyperparameter used to reweight classes and n_c is the number of samples in class c. We use default values of $\gamma = 2$ and $\beta = .5$.

As shown in Table 1, classifiers trained with focal losses perform as expected. Focal loss gives a higher per-class accuracy than vanilla softmax, and a higher mean accuracy than naively reweighted softmax. However, the accuracy of focal loss in absolute terms is still very low: both variants of focal loss appear to be navigating a simple tradeoff between the balanced and unbalanced baselines.

We conclude that training classifiers directly on the reference images is an ineffective approach to the meta-iNat benchmark. Classifiers must incorporate prior knowledge from the representation set, as there are too few reference images to train a discriminative classifier directly.

5.2 Representation Learning

We now focus on methods that learn features from the representation set. One approach to incorporating such a feature extractor is transfer learning: we train a classifier network on the representation set, but replace and re-train the final linear layer on the evaluation set. A number of techniques exist for obtaining this new classifier layer, forming a set of strong baselines that have been shown to outperform many prior few-shot learning approaches [17], [26], [58].

We implement these approaches on meta-iNat, shown in the middle rows of Table 1. SimpleShot [58] takes the feature extractor pretrained on the representation set and performs prototype-based classification with features either unnormalized (UN), or centered and L2-normalized (CL2N). This approach works significantly better than training from scratch, attaining top-1 accuracy scores over twice as high as classifiers not leveraging the representation set.

More sophisticated baselines improve upon these results. The Baseline and Baseline++ [26] models train new linear heads at test-time using SGD, and attain higher mean accuracy. However, they are slower at test-time and like the supervised approaches, fail to balance performance across classes. We fix this discrepancy via class-balanced implementations, where class size inversely reweights the classification loss used to retrain the linear head. Per-class accuracy for these models increases greatly.

The best performance overall is given by RFS-Simple [17], which uses an external multinomial regression solver to compute the new linear head. In contrast, prototypical networks (PN) trained episodically are disappointing. While they easily outperform the supervised baselines on meta-iNat, they underperform the transfer-learning baselines, which benefit from the larger set of reference images.

Interestingly, prototype-based classifiers (SimpleShot and PN) offer similar mean and per-class accuracies without label reweighting schemes or other class-balancing mechanisms. This suggests that prototypical networks, while providing no performance advantages over transfer learning in this heavy-tailed setting, are inherently class-balanced.

5.3 Our Techniques

Baseline prototypical network accuracy is disappointing. Batch folding, localization, and covariance pooling, however, improve accuracy significantly (Table 1, bottom rows), and swiftly overcome this initial shortfall in performance.

Batch folding: A prototypical network trained with batch folding outperforms the alternative by over a **3-point margin**, and already puts us within striking distance of the class-balanced transfer learning baselines. The per-class accuracy gain as a function of class size is plotted in Fig. 6. We see gains across the board, suggesting that batch folding does provide higher-quality gradients. At the same time, by incorporating more reference images during training, batch folding helps models generalize to larger classes: the

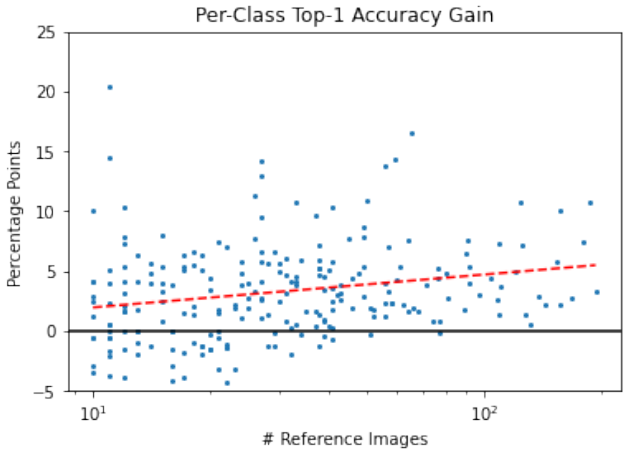

Fig. 6. Batch folding improves accuracy for all class sizes in expectation, but particularly helps with large ones ($r^2 = .05$). Gain is relative to a baseline prototypical network.

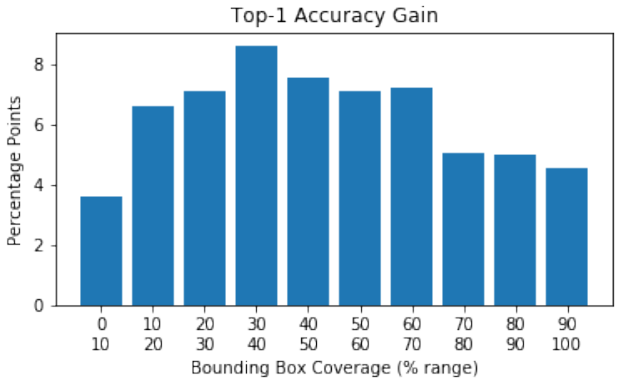

Fig. 7. Few-shot localization is more helpful on images with small regions of interest

positive slope of the best-fit line suggests that large classes benefit more from batch folding, though not at the expense of small ones.

Localization: Incorporating few-shot localization leads to another significant boost in performance, about **6 percentage points**. Note that 10% of reference images are annotated, amounting to only 1 to 20 images per category. This relatively cheap annotation has an outsize impact on performance. Interestingly, unsupervised localization provides a larger gain, about **8 percentage points**. We posit that few-shot localization underperforms its counterpart because it uses bounding boxes, a very coarse form of segmentation. Bounding boxes may include a significant amount of background, hurting the separation of foreground from background. Indeed, we find that localization particularly helps when objects are small, and bounding boxes cover less than half the image (Fig. 7). The decrease in gain for tiny objects is not entirely surprising - classification is inherently harder when the relevant object contains only a few pixels.

Covariance pooling: Accuracy improves yet again with covariance pooling, yielding a **4 point** gain over unsupervised localization and **9 points** over few-shot localization. Notably, covariance pooling causes class balance to break: large categories benefit disproportionately (Fig. 8).

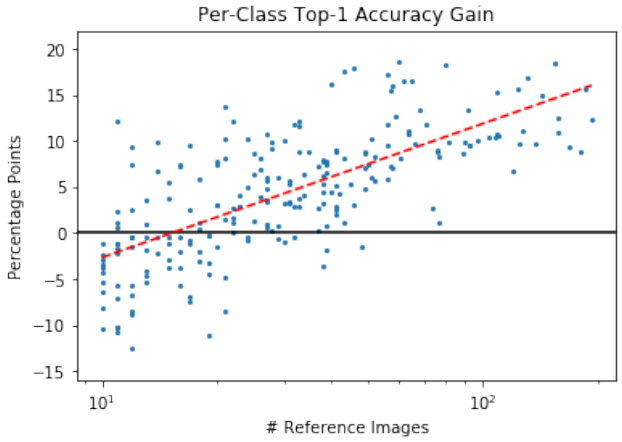

Fig. 8. Covariance pooling improves performance on large classes, at the expense of some small ones ($r^2 = .50$). Gain is relative to a batch folded network with few-shot localization and no covariance pooling.

We hypothesize that the high dimensionality of covariance space is responsible. Small categories do not provide enough reference images to span a sufficiently rich subspace, so centroid quality suffers.

Unsupervised localization does not interact well with covariance pooling, perhaps due to a similar dynamic during training. Each batch contains only 20 images per class, so covariance space may again be too high-dimensional for reference images to span. Thus the learned foreground and background vectors may overfit to a particular low-dimensional manifold on the representation set. Few-shot localization, which calculates these vectors dynamically, does not have this problem. We conclude that both localization techniques are useful for different settings.

Using all three techniques doubles the baseline prototypical network's top-1 accuracy. The best performer uses **batch folding, few-shot localization** and **covariance pooling**.

6 Analysis

Having demonstrated large gains for a single, small prototypical network, we now examine the robustness of these improvements, finding them to be highly stable. Batch folding, few-shot localization, and covariance pooling remain effective when applied to deeper, more sophisticated convolutional feature extractors. Through an ablation study, we observe that these techniques are also effective in any and all combinations: the gains associated with each are roughly independent. Finally, we show that few-shot localization, which requires bounding box annotations at test time, is surprisingly robust to the quantity. Performance saturates quickly, and we achieve notable improvement even with fewer annotations than classes.

6.1 Deeper Network Architectures

While batch folding, few-shot localization, and covariance pooling lead to substantial improvement in Table 1, accuracy is still low in general. For more powerful models, these improvements might shrink or even disappear entirely. Thus, we re-evaluate our three techniques using a larger

Model	Top-1 Accuracy		Top-5 Accuracy	
	Mean	Per-Class	Mean	Per-Class
PN	37.03±.21	37±.33	70.75±.39	69±.39
PN+BF	41.30±.17	41±.22	73.60±.12	72±.22
PN+BF+usL	44.68±1.18	43±1.04	74.59±1.03	73±1.03
PN+BF+fsL	45.45±.05	45±.07	75.8±.05	74±.05
PN+BF+usL+CP	27.47±15.15	26±13.62	53.24±17.55	51±16.73
PN+BF+fsL+CP	**51.56±.06**	**48±.03**	**78.83±.05**	**76±.05**

TABLE 2
Accuracy with the ResNet-12 model, with 95% confidence intervals from 4 trials. Models (and except for usL, results) are as in Table 1.

	PN+BF			PN+BF+usL+CP		
	Train Acc (20-way)	Train Loss	Test Acc (all-way)	Train Acc (20-way)	Train Loss	Test Acc (all-way)
Model 1	94.98	13.29	41.31	97.54	6.33	27.24
Model 2	94.85	13.62	40.85	97.65	6.05	10.28
Model 3	94.90	13.60	40.71	97.67	6.01	19.68
Model 4	94.92	13.57	40.97	97.75	5.88	48.00
Average	94.91	13.52	**40.96**	**97.65**	**6.07**	26.30

TABLE 3
Training and testing performance metrics for ResNet-12 models without and with unsupervised localization and covariance pooling. Severe overfitting is evident for the latter, though the degree varies widely between training runs (Model 4 for example generalizes well).

Model	Top-1 Accuracy		Top-5 Accuracy	
	Mean	Per-Class	Mean	Per-Class
000*	$16.07 \pm .19$	$17.55 \pm .19$	$42.10 \pm .21$	$41.98 \pm .18$
100*	$20.04 \pm .04$	$20.81 \pm .08$	$47.86 \pm .31$	$46.57 \pm .23$
010	$21.55 \pm .09$	$22.37 \pm .08$	$50.20 \pm .08$	$48.70 \pm .08$
001	$24.31 \pm .48$	$24.64 \pm .39$	$53.29 \pm .76$	$51.09 \pm .63$
020	$24.32 \pm .15$	$24.68 \pm .81$	$53.16 \pm .89$	$51.44 \pm .94$
021	25.60 ± 1.02	$25.51 \pm .90$	54.66 ± 1.07	$52.04 \pm .91$
110*	$26.25 \pm .05$	$26.29 \pm .04$	$55.43 \pm .09$	$53.01 \pm .08$
011	$27.46 \pm .15$	$26.39 \pm .14$	$56.37 \pm .17$	$52.37 \pm .17$
120*	$28.75 \pm .13$	$28.39 \pm .15$	$57.90 \pm .24$	$55.27 \pm .37$
101	$31.06 \pm .61$	$29.07 \pm .53$	$60.41 \pm .67$	$55.50 \pm .62$
121*	$32.74 \pm .13$	$30.52 \pm .13$	$61.32 \pm .14$	$56.62 \pm .16$
111*	$35.52 \pm .05$	$31.69 \pm .06$	$63.76 \pm .09$	$57.33 \pm .10$

TABLE 4
Ablation results, models sorted by top-1 accuracy. Three-digit model names indicate the presence or absence of batch folding, localization, and covariance pooling, in that order. For example, '101' indicates a model with batch folding and covariance pooling, but no localization. We use '0' to indicate no localization, '1' for few-shot localization, and '2' for unsupervised localization. * model present in Table 1.

ResNet-12 network. The ResNet-12 architecture consists of three residual blocks, with three convolutional layers on the residual stem and one on the main stem. For more details, refer to [21]. Batch folding, few-shot localization and unsupervised localization all work as before. For covariance pooling, however, the output dimensionality of ResNet-12 (512) is too large for the resulting $512 \times 512 = 262144$-dimensional embeddings to train stably or efficiently. We therefore add an additional down-projection layer to 128 dimensions before performing covariance pooling. Results can be found in Table 2.

The results on ResNet-12 largely mirror the results for the four-layer architecture. The performance gains provided by batch folding, few-shot localization, and covariance pooling remain significant and produce the best available model. Applying batch folding outperforms the baseline in Table 2 by about a **4-point margin**, comparable to the four-layer results. Incorporating few-shot localization into ResNet-12 further improves performance by about **4 percentage points**, and accuracy again improves with covariance pooling, yielding a **6 point gain**. While smaller than the corresponding four-layer gains, these results are still encouraging, as roughly 2/3 of each gain is preserved over a baseline that has greater than twice the top-1 accuracy of its four-layer counterpart. Batch folding, few-shot localization, and covariance pooling thus continue to function as desired when applied to more powerful feature extractor backbones.

One notable departure from the four-layer results is the weakness of both models employing unsupervised localization. Unlike the four-layer architecture, unsupervised localization underperforms few-shot localization by a small quantity, while unsupervised localization with covariance pooling fails completely, underperforming even the baseline prototypical network by a large margin. Given the larger output dimensionality of the ResNet-12 backbone network, we found that the parametrized foreground/background vectors of unsupervised localization allow these models to overfit to the representation set. The overfitting is made much worse by the addition of covariance pooling, where output vectors may fall onto a very specific manifold that the localizer cannot gracefully handle departures from.

We illustrate this in Table 3: training accuracy/loss becomes significantly higher/lower with the addition of unsupervised localization and covariance pooling. Meanwhile, test time performance on novel classes becomes much worse, demonstrating severe overfitting. These results also explain the unusually large confidence intervals for the unsupervised localization model with covariance pooling in Table 2 (fifth row). The degree of overfitting varies widely between individual runs: training metrics in Table 3 are roughly equally uniform across models in each setting, while up to 37 points of test-time accuracy separate the best-performing localization model from the worst. Though overfitting occurs, the exact degree is inconsistent.

We conclude that unsupervised localization clearly does not generalize to larger architectures. The gains from our other techniques, however, remain significant, and do not disappear as baseline performance improves.

6.2 Ablation

Batch folding, few-shot localization, and covariance pooling produce successively better few-shot classifiers, but it is not clear whether an arbitrary combination is beneficial. To show that these techniques are truly mutually beneficial, we perform an ablation study on the four-layer network architecture with results given in Table 4.

Between any two models with different numbers of techniques, the one with more always outperforms the one with less. Two conclusions can be drawn: first, batch folding, localization, and covariance pooling consistently improve performance on meta-iNat. Using each technique is always better than not. Second, the gains from each technique are roughly comparable across settings: in no case does a single technique produce a larger gain than the other two combined. We therefore conclude that the contributions of

Localizer	% annotation	Mean acc.	Per-Class acc.
Untrained	10%	19.74±.03	20.42±.06
No Gradient	10%	22.77±.23	22.86±.18
Jointly trained	Supercategory	23.67±.79	24.08±.66
	1%	25.85±.11	25.96±.09
	4%	26.17±.08	26.22±.06
	16%	**26.28±.05**	**26.30±.04**
	64%	26.21±.04	26.25±.03

TABLE 5
Top-1 accuracy with 95% confidence intervals for few-shot localization models as annotations increase, with comparison to baseline localizers. All models use batch folding.

Model	Four-Layer Accuracy	ResNet-12 Accuracy
PN	67.55±.46	87.25±.3
PN+BF	69.15±.42	87.23±.33
PN+BF+fsL	**71.79±.46**	**88.09±.31**
PN+BF+fsL+CP	68.24±.47	87.91± .33

TABLE 6
Results (accuracy with 95% confidence intervals) for the four-layer and ResNet-12 classifiers under 5-way/5-shot episodic evaluation. Models are as in Table 1.

batch folding, localization, and covariance pooling are both consistent and roughly independent.

6.3 Analyzing Few-Shot Localizer Behavior

Finally, we evaluate the number of bounding box annotations needed for good classification accuracy. As shown in Table 5, performance saturates at 16% bounding box availability, but even at 1% (amounting to one box per class), performance decreases only slightly. This scarcity can go further: categories in meta-iNat are grouped into nine supercategories, so we also try using one box per supercategory, nine total. Accuracy does drop significantly, but is still better than models that do not localize at all. Thus localization can lead to real accuracy gains using hardly any annotations at all, to our knowledge a first-of-its-kind finding.

Joint training: Although the few-shot localizer never receives direct training supervision, it must still be learned jointly with the classifier. Table 5 also compares localizers that are not jointly trained. Applying few-shot localization to a network trained without it leads to a drop in performance ("Untrained"). Training the network to use localization, but preventing backpropagation through the localizer itself by treating the foreground/background vectors as constants, also leads to a drop in performance ("No Gradient"). Localization thus provides a useful training signal, but must itself be trained with the classifier for maximum benefit.

7 Generalization

Moving beyond the proposed evaluation on meta-iNat, we investigate four new settings. To test robustness over evaluation methods, we re-evaluate meta-iNat performance using the standard 5-way/5-shot episodic testing scheme. To test generalization over domain shift, we create a second split of meta-iNat based on supercategories, similar to the setup of tiered ImageNet [31]. We incorporate pretrained ResNet-50 feature extractors, to show that our techniques generalize to more sophisticated training schemes. Finally, these techniques are tested on CUB [29], a benchmark from prior literature. With some expected caveats involving small shot numbers, our results generalize well to all settings.

7.1 Episodic Evaluation

We have argued that consistently confining the way and shot of the evaluation task to five is unrealistic, as this does not reflect the wide variety and long-tailed distribution of classes in the wild. However, performance in this setting may still be of interest. We therefore examine the effects of batch folding, localization and covariance pooling on four-layer and ResNet-12 few-shot classifiers when evaluated using five-way/five-shot episodes. The training process is unchanged. Results are presented in Table 6. Given the failure of unsupervised localization in ResNet-12 (Table 2), we do not consider that particular technique.

For both architectures, batch folding and few-shot localization together yield the highest accuracy. The slight drop in performance from covariance pooling is disappointing but not entirely surprising. Five support points is simply too few to span the more expansive feature space, resulting in poor-quality class centroids. This is corroborated by the sharply positive trendline in Fig. 8: covariance pooling produces consistent gains for large classes at the expense of small ones. 5-way/5-shot episodes consist entirely of small classes, so this drop is to be expected. Meanwhile, batch folding and few-shot localization remain effective, albeit by smaller margins than in the non-episodic evaluation task.

7.2 Tiered meta-iNat

We also wish to evaluate our results in settings where transfer learning is more difficult, and switching from the representation set to the evaluation set involves substantial domain shift. While the representation sets and reference/query evaluation sets of meta-iNat do contain distinct classes, the overall distribution of fish/birds/moths/etc. is the same, so it could be argued that meta-iNat actually involves very little *semantic* domain shift, and most of the challenge lies in reassigning labels to classes. To introduce semantic domain shift, we construct a new version of meta-iNat in the spirit of tiered ImageNet [31], which we call Tiered meta-iNat. Rather than assign categories to the representation and evaluation sets randomly, we instead split by supercategory. Insects and arachnids (354 total) form the evaluation set, while everything else (birds, fish, mammals, reptiles, etc.) comprises the representation set. Training and evaluation are performed as in the standard meta-iNat, with results given in Table 7.

Transfer learning on Tiered meta-iNat is much harder than in the original setting. Scores are uniformly lower across the board. However, overall trends remain exactly the same. Batch folding outperforms standard prototypical networks and transfer learning baselines by **2 points**. Few-shot and unsupervised localization lead to similar, substantial accuracy gains (**4 points**). Covariance pooling also improves (**5 points**), but again causes mean accuracy to outstrip per-class accuracy. Unsupervised localization routinely

Model	Top-1 Accuracy		Top-5 Accuracy	
	Mean	Per-Class	Mean	Per-Class
Reweighted Softmax	4.59±.21	5.38±.22	15.95±.58	16.57±.53
Transfer Learning	6.34±.23	6.19±.14	18.89±.48	17.86±.49
PN	5.33±.18	6.31±.18	17.41±.45	18.32±.27
PN+BF	7.29±.11	8.24±.13	22.09±.35	22.53±.37
PN+BF+fsL*	11.69±.06	12.38±.07	30.64±.11	29.86±.09
PN+BF+usL	12.46±.59	12.95±.51	32.28±1.1	31.18±.95
PN+BF+usL+CP	17.65±.21	16.72±.18	40.16±.26	36.19±.48
PN+BF+fsL+CP*	**20.02±.13**	**17.32±.09**	**43.45±.20**	**36.65±.15**

TABLE 7
Results on the Tiered meta-iNat benchmark, with 95% confidence intervals. Models are as in Table 1.

Model	Top-1 Accuracy		Top-5 Accuracy	
	Mean	Per-Class	Mean	Per-Class
Transfer Learn (top)	19.27±.17	18.72±.20	44.02±.30	41.2±.36
Transfer Learn (full)	22.52±.58	18.22±.40	48.16±.60	40.38±.48
PN	35.35±.24	35.59±.11	67.82±.13	66.33±.19
PN+BF	37.36±.15	36.73±.12	69.25±.16	67.03±.15
PN+BF+fsL*	46.2±.04	44.43±.08	75.87±.04	**73.26±.06**
PN+BF+fsL+CP*	**51.25±.13**	**46.04±.13**	**77.5±.06**	72.14±.05

TABLE 8
Results on meta-iNat using ResNet50 features, with 95% confidence intervals. Transfer Learning (top) adjusts unfrozen upper layers on reference images, while (full) fine-tunes the entire network. Other models are as in Table 1.

underperforms few-shot localization when using covariance pooling, so we remove it from future tests.

7.3 Pretrained Features

We have shown that batch folding, few-shot localization, and covariance pooling are successful at improving the performance of classifiers trained using those techniques. However, in many cases it is advantageous to use a freely available feature extractor pretrained on another task, such as ImageNet. It is therefore of interest whether or not our techniques can also improve a classifier with a frozen, pretrained feature extractor. To that end, we replace the bottom two layers of our networks with a ResNet-50 pretrained on ImageNet. We extract feature maps from the second stage of the network, resulting in a final feature map size of $14 \times 14 \times 64$. Classification, batch folding, localization, and covariance pooling work as before, and training procedure is the same, except that we freeze the ResNet components and train only the top two layers. Further details can be found in Appendix A, and results are presented in Table 8.

Using the pretrained ResNet-50 model, it is possible to perform transfer learning directly from ImageNet to the meta-iNat evaluation set. Freezing the ResNet, and training just the top two layers on reference images, works poorly given the power of the model. Fine-tuning the entire network on reference images works slightly better, but introduces disparity between the mean and per-class accuracy. Freezing the ResNet and training the top layers as a prototypical network improves top-1 accuracy by **13 percentage points**. Batch folding, few-shot localization, and covariance pooling provide another **16 points**. These results are roughly comparable to the ResNet-12 performance in Table 2, though in this case we are only training two layers, rather than the entire, much larger, network.

Model	Top-1 Accuracy	Top-5 Accuracy
PN	30.26±0.55	66.74±1.06
PN+BF	30.18±0.30	65.37±0.59
PN+BF+fsL	36.90±0.70	73.63±0.53
PN+BF+fsL+CP	**44.33±1.38**	**78.18±0.77**

TABLE 9
All-way all-shot accuracy on CUB with 95% confidence intervals over 4 random trials.

Interestingly, there remains little saturation in performance as the feature extractor improves. The 16-point gain in top-1 accuracy over ResNet-50 prototypical networks (Table 8) is comparable to both the 14.5-point gain for ResNet-12 (Table 2) and the 19.5- and 15-point gains for four-layer architectures (Tables 1 and 7). This is a stark contrast to prior work, which demonstrates a vanishing contribution from few-shot learning techniques as architectures improve [26]. We conclude that our techniques are helpful across a range of architectures and training schemes.

7.4 CUB

Batch folding, few-shot localization, and covariance pooling improve accuracy on large evaluation sets with long-tailed class distributions. To see if these techniques still help with smaller, more evenly-distributed few-shot learning problems, we also experiment with the CUB [29] dataset, which consists of 11,788 images from 200 classes. In this benchmark the classifier must distinguish different species of birds. Using the same split as [59], the categories are divided into 100 base, 50 validation and 50 novel classes. For each validation and novel class, 20% of its images are used as the reference set and the remaining images form the query set. Results for all-way evaluation are shown in Table 9.

An immediate departure from prior results is the fact that batch folding no longer improves performance. This is likely due to the small scale of the dataset and the limited testing shot number (the CUB reference set contains only 8-12 images per class, whereas batch folding mimics 20-shot training episodes). Few-shot localization and covariance pooling, however, continue to greatly improve accuracy.

8 Conclusion

In this paper, we have shown that past work on classical or few-shot balanced benchmarks fails to generalize to realistic heavy-tailed classification problems. We show that parameter-free localization from limited bounding box annotations, and improvements to training and representation, provide large gains beyond those previously observed in data abundant settings. Ours is but a first step in addressing broader questions of class balance and data scarcity.

Appendix A
Network Architectures

Our learner architectures mimic the original prototypical networks. Networks contain four 64-channel 3×3 convolutional layers, with Batchnorm, ReLU, and 2×2 max-pooling in between. This is followed by Batchnorm and 10×10 global average-pooling, for a 64-dimensional feature vector. Unlike prototypical networks, which flatten the feature

map to a 6400-dimensional vector, we use average pooling to eliminate spatial priors on uncentered, uncropped images. Localization and covariance pooling models replace average-pooling layers with the appropriate algorithm(s). Baselines with linear heads use ReLU before global-pooling.

Models based on ResNet-12 follow the standard architecture: an initial 3×3 convolution to 32 channels, followed by four residual blocks. Each block doubles the channel size and contains three 3×3 convolutional layers on the residual stem and one layer on the main stem, with Batchnorm and ReLU interspersed as appropriate. 2×2 max-pooling decreases the spatial resolution between residual blocks.

Models based on ResNet-50 use the first two stages of a ResNet-50 model, pretrained on ImageNet, to produce 28×28 feature maps with 512 channels. Learned layers consist of 2×2 max-pooling, 3×3 convolution to 128 channels, Batchnorm, ReLU, a second convolution to 64 channels, and Batchnorm. The resulting 14×14 feature maps are localized, average pooled, or covariance pooled as appropriate.

APPENDIX B
TRAINING AND IMPLEMENTATION DETAILS

B.1 Training

We train meta-iNat models via SGD with Adam, using an initial learning rate of 10^{-3} and dividing by two every epoch. Each epoch consists of 10 passes over the representation set, with five epochs in total. We unit-normalize input color channels and use random horizontal flipping but no other data augmentation. Because bounding boxes are downsized to very small resolutions (e.g. 10×10), we allow for float-valued masks representing the percentage of each grid location contained within a full-resolution box.

B.2 Batch Sampling

While prior work uses random batch sampling with replacement during training and testing, meta-iNat uses a different sampling procedure for each. During each training iteration, classes are sampled without replacement: so long as classes have sufficient remaining images to create a batch, they are sampled proportionately to the number of available images they contain. This ensures that differently-sized classes occur at constant, representative rates throughout the entire sampling process. Images are then selected from the sampled classes, and those images are subsequently unavailable for further training until the next pass over the dataset. Episodic evaluation uses the same procedure.

During non-episodic testing, we wish to evaluate classification accuracy for relatively large numbers of classes and images. For sufficiently high values, it becomes impossible to process the sampled datasets as single batches in computer memory. Category sizes also vary so widely that it no longer makes sense to use constant sample sizes. Rather than attempt to evaluate a given network on a large number of large and complicated datasets, we instead impose a single reference/query split over the evaluation set.

Evaluation consists of one pass over the reference images followed by one pass over the query images. During the reference pass, each category is split into manageable batches, and the class centroid is computed from a running total of the embedding vectors. The query pass is divided the same way, and the pre-computed centroids are used to make class predictions. When localization is used, we run 10 trials where different 10% subsets of the reference images are randomly selected to receive bounding box annotations.

The above applies to prototype-styled classifiers, which require representation and query sets for a limited selection of classes in every batch. For baseline softmax classifiers, we instead train using random sampling without replacement, and a batch size of 128. Annealing schedule is as above.

ACKNOWLEDGMENTS

This work was partly funded by a grant from Aricent and the DARPA Learning with Less Labels program (HR001118S0044).

REFERENCES

[1] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, 2017.
[2] B. Hariharan and R. Girshick, "Low-shot learning by shrinking and hallucinating features," in *IEEE International Conference on Computer Vision*, 2017.
[3] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *Neural Information Processing Systems*, 2017.
[4] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Neural Information Processing Systems*, 2016.
[5] G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. J. Belongie, "The inaturalist species classification and detection dataset," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
[6] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *European Conference on Computer Vision*, 2012.
[7] P. Koniusz, F. Yan, P.-H. Gosselin, and K. Mikolajczyk, "Higher-order occurrence pooling for bags-of-words: Visual concept detection," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 2, 2017.
[8] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," in *Transactions on Pattern Analysis and Machine Intelligence*, 2017.
[9] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, 2017.
[10] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-shot learning for semantic segmentation," in *British Machine Vision Conference*, 2017.
[11] A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations*, 2019.
[12] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.
[13] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017.
[14] H. Edwards and A. Storkey, "Towards a neural statistician," in *International Conference on Learning Representations*, 2017.
[15] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
[16] L. Bertinetto, J. Henriques, P. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *International Conference on Learning Representations*, 2019.
[17] Y. Tian, Y. Wang, D. Krishnan, J. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: a good embedding is all you need?" in *European Conference on Computer Vision*, 2020.
[18] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[19] H. Zhang and P. Koniusz, "Power normalizing second-order similarity network for few-shot learning," in *IEEE Winter Conference on Applications of Computer Vision*, 2019.

[20] Y. xiong Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[21] B. N. Oreshkin, P. R. López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *Neural Information Processing Systems*, 2018.

[22] H. Ye, H. Hu, D. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[23] M. Bauer*, M. Rojas-Carulla*, J. B. Świątkowski, B. Schölkopf, and R. E. Turner, "Discriminative k-shot learning using probabilistic models," in *Second Workshop on Bayesian Deep Learning at the 31st Conference on Neural Information Processing Systems*, 2017, *equal contribution.

[24] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based meta-learning as hierarchical bayes," in *International Conference on Learning Representations*, 2018.

[25] C. Zhang, Y. Cai, G. Lin, and C. Shen, "Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[26] W. Chen, Y. Liu, Z. Kira, Y. F. Wang, and J. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[27] Y. Chen, X. Wang, Z. Liu, H. Xu, and T. Darrell, "A new meta-baseline for few-shot learning," *arXiv:2003.04390*, 2020.

[28] R. Hou, H. Chang, B. Ma, S. Shan, and X. Chen, "Cross attention network for few-shot classification," in *Neural Information Processing Systems*, 2019.

[29] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," California Institute of Technology, Tech. Rep., 2011.

[30] B. Lake, R. Salakhutdinov, and J. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, 2015.

[31] M. Ren, S. Ravi, E. Triantafillou, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," in *International Conference on Learning Representations*, 2018.

[32] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[33] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[34] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *European Conference on Computer Vision*, 2014.

[35] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[36] B. Zhou, À. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Neural Information Processing Systems*, 2014.

[37] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Department of Computer Science, University of Toronto, 2009.

[39] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[40] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European Conference on Computer Vision*, 2010.

[41] D. Acharya, Z. Huang, D. P. Paudel, and L. V. Gool, "Covariance pooling for facial expression recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[42] A. R. Chowdhury, T. Lin, S. Maji, and E. Learned-Miller, "One-to-many face recognition with bilinear cnns," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[43] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? - weakly-supervised learning with convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[44] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[45] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[46] N. Araslanov and S. Roth, "Single-stage semantic segmentation from image labels," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[47] M. Guillaumin, D. Küttel, and V. Ferrari, "Imagenet auto-annotation with segmentation propagation," *International Journal of Computer Vision*, vol. 110, no. 3, 2014.

[48] D. Kuettel, M. Guillaumin, and V. Ferrari, "Segmentation propagation in imagenet," in *European Conference on Computer Vision*, 2012.

[49] C. Doersch, A. Gupta, and A. Zisserman, "Crosstransformers: spatially-aware few-shot transfer," in *Neural Information Processing Systems*, 2020.

[50] D. Papadopoulos, J. Uijlings, F. Keller, and V. Ferrari, "Extreme clicking for efficient object annotation," in *IEEE International Conference on Computer Vision*, 2017.

[51] "Broadcasting - numpy v1.15 manual," https://docs.scipy.org/doc/numpy-1.15.0/user/basics.broadcasting.html, accessed: 2018-11-16.

[52] "Broadcasting semantics - pytorch master documentation," https://pytorch.org/docs/stable/notes/broadcasting.html, accessed: 2018-11-16.

[53] "Broadcasting semantics — xla — tensorflow," https://www.tensorflow.org/xla/broadcasting, accessed: 2018-11-16.

[54] N. Otberdout, A. Kacem, M. Daoudi, L. Ballihi, and S. Berretti, "Deep covariance descriptors for facial expression recognition," in *British Machine Vision Conference*, 2018.

[55] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *European Conference on Computer Vision*, 2004.

[56] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision*, 2017.

[57] Y. Cui, M. Jia, T. Lin, Y. Song, and S. J. Belongie, "Class-balanced loss based on effective number of samples," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[58] Y. Wang, W.-L. Chao, K. Weinberger, and L. van der Maaten, "Simpleshot: Revisiting nearest-neighbor classification for few-shot learning," *arXiv:1911.04623*, 2019.

[59] L. Tang, D. Wertheimer, and B. Hariharan, "Revisiting pose-normalization for fine-grained few-shot recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Davis Wertheimer received his Bachelor's degree in Symbolic Systems with distinction from Stanford University, and is currently a Ph.D. candidate in Computer Science at Cornell University. His research interests are in computer vision, few-shot classification, and learning from limited data. Bharath Hariharan is his advisor.

Luming Tang received his Bachelor's degree in Mathematics and Physics from Tsinghua University, and is currently a Ph.D. student in Computer Science at Cornell University, advised by Prof. Bharath Hariharan.

Dhruv Baijal received his Bachelor's and Master's degree in Computer Science from Cornell University. He is currently working at Adobe.

Pranjal Mittal received their Bachelor's Degree in Computer Science from Cornell University. He is currently working at Oracle Cloud Infrastructure as a software engineer and graduated Summa Cum Laude with Honors from Cornell University.

Anika Talwar received her Bachelor's Degree in Computer Science from Cornell University and graduated Magna Cum Laude with Honors. She is from Boston, MA and is currently working at Facebook as a Software Engineer.

Bharath Hariharan is an assistant professor in Cornell University. He was previously a postdoctoral researcher at Facebook AI Research. He received his PhD from University of California, Berkeley under the tutelage of Prof. Jitendra Malik. His research focuses on training computer vision systems with very limited training data.

Summary of Differences

An earlier version of this manuscript was published in the Computer Vision and Pattern Recognition (CVPR) conference in 2019, titled "Few-Shot Learning with Localization in Realistic Settings". This journal submission contains substantial extensions and improvements, summarized as follows:

- **Results on meta-iNat using ResNet-12**: The original CVPR paper tested only simple 4-layer architectures for the main set of results on meta-iNat. Since then, the field has shifted toward stronger classifier architectures. We therefore reproduce our experiments using the standard ResNet-12 network backbone and incorporate these results into a new Analysis section.

- **Two additional supervised meta-iNat baselines**: The original CVPR paper considered only very basic supervised models on the meta-iNat reference images as a sanity check. This included two naive approaches to handling class imbalance (reweighted loss and resampled data). We now incorporate two more recent and sophisticated techniques for handling class imbalance: focal loss [56] and class-balanced focal loss [57]. This provides a stronger set of supervised baselines for comparison.

- **Seven additional few-shot meta-iNat baselines**: Prototypical networks are the only few-shot baseline considered in the original CVPR paper, as most few-shot learning techniques at the time assumed you could fit the entire support set into memory (rather than relying solely on aggregate statistics), which is not true for the all-way/all-shot evaluation in meta-iNat. While this is still the case in more recent work (i.e. LEO [11], FEAT [22], and CrossTransformer [49] all rely on this assumption), recently proposed "strong baseline" models based on retrained linear heads are feasible. We therefore replace the original heuristic "transfer learning" baseline with seven new baselines from recent prior literature [17], [26], [58]. Like prior literature, we find that these models handily outperform straightforward prototypical networks, though our techniques continue to provide superior performance by a large margin.

- **Experiments on CUB**: To further test the generalization ability of our techniques, we evaluate them on the standard few-shot CU-Birds benchmark in a new Generalization section. Similarities to and differences from the meta-iNat results are discussed.

- **Margin implementation for batch folding**: We provide a mathematical derivation for a more efficient version of batch folding, implemented as a margin in the predicted logit scores. This implementation is much simpler and more straightforward, as it no longer requires calculating a unique set of centroids for each predicted data point.

- **Episodic evaluation**: The original CVPR paper attempted to examine how our techniques generalize to limited-way/few-shot settings by evaluating on a variant of mini-ImageNet. However, our variant dataset did not truly reproduce mini-ImageNet, as we could only include classes with bounding box annotations available. This made these results difficult to reproduce, interpret, and compare to other experiments. We therefore replace them with direct 5-way/5-shot evaluation on meta-iNat, using both four-layer and ResNet-12 backbones, for a clearer and more informative set of episodic evaluation results.

- **Organizational changes**: The results section has been reworked and expanded. Instead of a single results section, we now provide 1) an Experiments section strictly for the main meta-iNat results, 2) an Analysis section, which contains the new ResNet-12 results, the old localizer behavior results, and the full ablation study for batch folding, localization and covariance pooling, and 3) a Generalization section for other training, evaluation, and benchmark settings.

- **General text changes and updates**: Language has been updated throughout for clarity and topicality. The Supercategory meta-iNat class split ("tiered meta-iNat") now follows the naming convention of tiered mini-ImageNet [31], prior work has been updated, the explanation of batch folding within few-shot localization has been clarified, etc.